

TAG - Módulo 3

Desafio Formativo

TUTORIAL (Parte II)

Enunciado

Fazer um mini-jogo em que:

- Temos um **círculo controlado pelos cursores**
- Temos outro **círculo a fazer springing** de um lado para o outro
- O jogador **perde** quando o círculo controlado pelos cursores **colide** com o círculo que está a fazer springing de um lado para o outro.
- O jogador **ganha** se conseguir **sair do canvas** sem acertar no outro círculo.

Tutorial parte I

Fazer um mini-jogo em que:

- Temos um **círculo controlado pelos cursores**



- Temos outro **círculo a fazer springing** de um lado para o outro



- O jogador **perde** quando o círculo controlado pelos cursores **colide** com o círculo que está a fazer springing de um lado para o outro.

- O jogador **ganha** se conseguir **sair do canvas** sem acertar no outro círculo.

O jogador ganha

Quando há **colisão** do círculo (ball) com os **limites do canvas**.

Podemos utilizar código de outras aulas! Vamos ao código modulo4-3, e vamos usar e modificar a função `checkWalls()`. Vamos copiá-la para o nosso código, antes da função `drawFrame()`:

```
function checkWalls (ball) {  
  if(ball.x + ball.radius > canvas.width) {  
    ball.x = canvas.width - ball.radius;  
    ball.vx *= bounce;  
  }  
  else if (ball.x - ball.radius < 0) {  
    ball.x = ball.radius;  
    ball.vx *= bounce;  
  }  
  
  if(ball.y + ball.radius > canvas.height) {  
    ball.y = canvas.height - ball.radius;  
    ball.vy *= bounce;  
  }  
  else if (ball.y - ball.radius < 0){  
    ball.y = ball.radius;  
    ball.vy *= bounce;  
  }  
}
```

Mas não queremos que o círculo volte para trás,
Apenas verificar se houve uma colisão com as paredes.
Assim, vamos modificar...

Modificar checkWalls()

É esta
condição
que nos
interessa

```
function checkWalls (ball) {  
  if(ball.x + ball.radius > canvas.width) {  
  }  
  else if (ball.x - ball.radius < 0) {  
  }  
  if(ball.y + ball.radius > canvas.height) {  
  }  
  else if (ball.y - ball.radius < 0){  
  }  
}
```

O círculo sai do canvas pela direita

O círculo sai do canvas pela esquerda

O círculo sai do canvas por baixo

O círculo sai do canvas por cima

Então vamos fazer acontecer alguma coisa neste caso.

Ou seja, adicionar

console.log("You WIN!");

```
if(ball.y + ball.radius > canvas.height) {  
  console.log("You WIN!");  
}
```

Not so fast though...

Não precisamos de todas as condições. Na realidade, nos outros limites não acontece nada. Vamos limpar o código.

```
/* since you need the ball to vertically cross the canvas  
to win, this function only checks whether we have reached the  
maximum height of the canvas */  
function checkWalls (ball) {  
  if(ball.y + ball.radius > canvas.height) {  
    console.log("You WIN!");  
  }  
}
```

Agora a função checkWalls() é muito mais simples.

Vamos testar...

Ainda não imprime nada...




Porquê?

Ainda não imprime nada...

Já repararam que ainda não chamamos a função `checkWalls()`?

Vamos acrescentar, no fim da função `drawFrame()`, a invocação à função.



```
ball.draw(context);

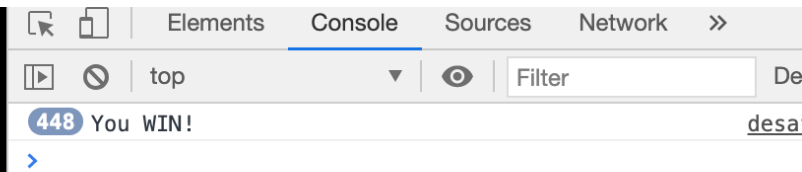
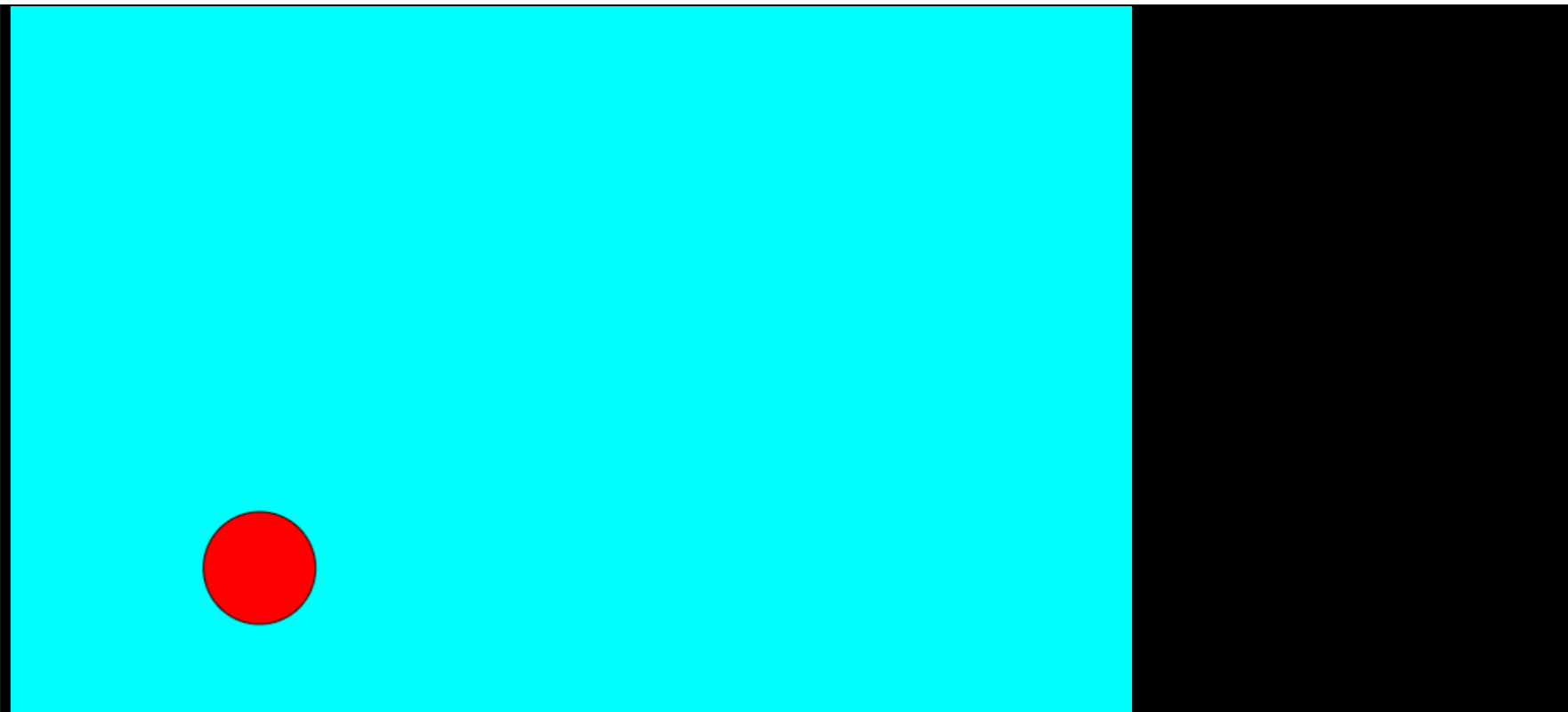
var dx = targetX - ball2.x,
    a2x = dx * spring;

v2x += a2x;
ball2.x += v2x;

checkWalls(ball);
ball2.draw(context);
```

E fazer reload no browser.

Et voilà!



Yay!

Enunciado

Fazer um mini-jogo em que:

- Temos um **círculo controlado pelos cursores**



- Temos outro **círculo a fazer springing** de um lado para o outro



- O jogador **perde** quando o círculo controlado pelos cursores **colide** com o círculo que está a fazer springing de um lado para o outro.

- O jogador **ganha** se conseguir **sair do canvas** sem acertar no outro círculo.



O jogador perde

Quando há **colisão** do círculo (ball) com o outro círculo (ball2).

Vamos desta vez fazer o código, embora também já o tenhamos de outra aula.

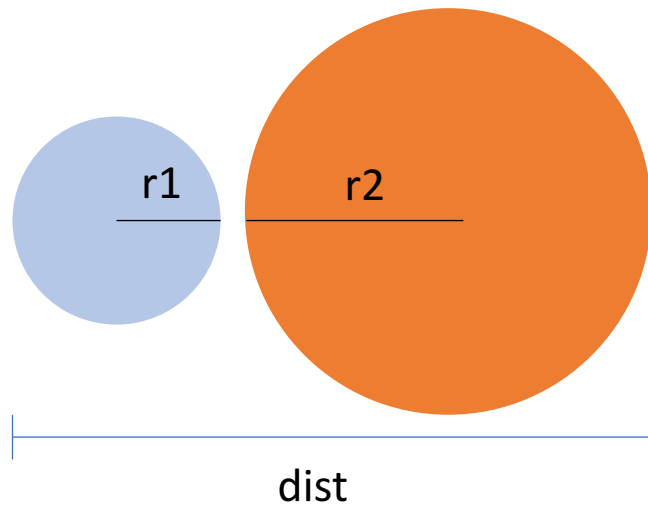
Vamos então criar uma nova função checkCollision(), antes da função checkWalls().

Esta nova função vai
verificar se existe
colisão entre
quaisquer dois círculos

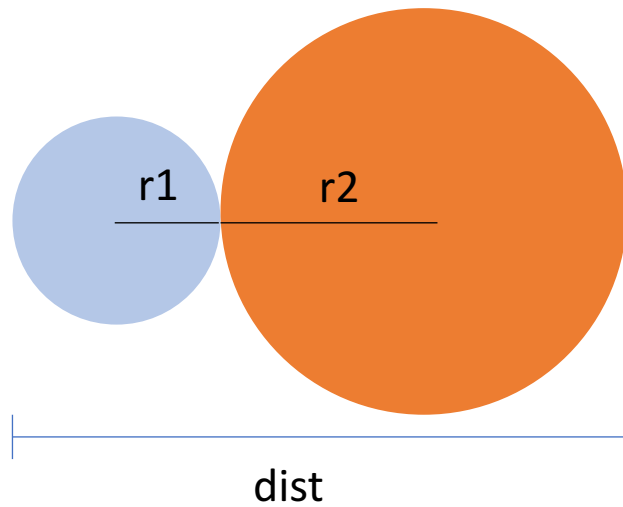
```
function checkCollision(ball1, ball2) {  
  
}  
  
/* since you need the ball to vertically cross the canvas  
to win, this function only checks whether we have reached the  
maximum height of the canvas */  
function checkWalls (ball) {  
  if(ball.y + ball.radius > canvas.height) {  
    //console.log("You WIN!");  
  }  
}  
}
```

Lógica da Colisão

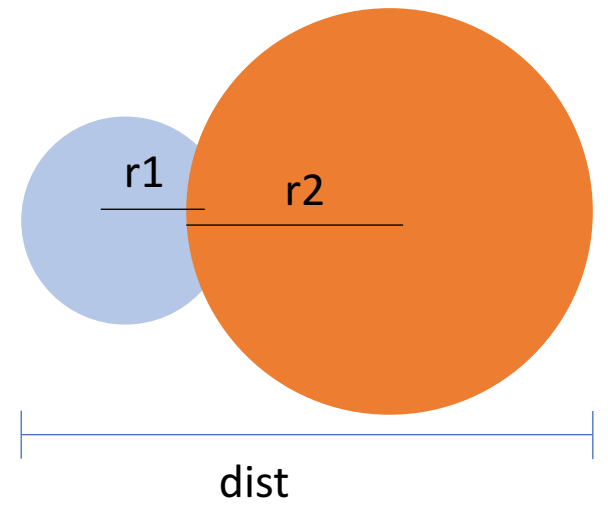
Qual a lógica que de uma colisão entre dois círculos? Se a distância entre ambos for igual ou inferior à soma dos seus raios. Lembram-se? Ou seja:



$$dist > r_1 + r_2$$



$$dist = r_1 + r_2$$



$$dist < r_1 + r_2$$

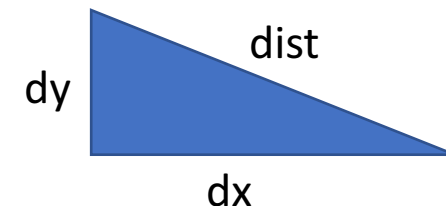
Então vamos preencher...

Quando a distância entre os círculos é igual à soma dos raios, já temos colisão.

Não se esqueçam que temos de ver a distância em x e em y, e depois aplicar o *teorema de Pitágoras*:

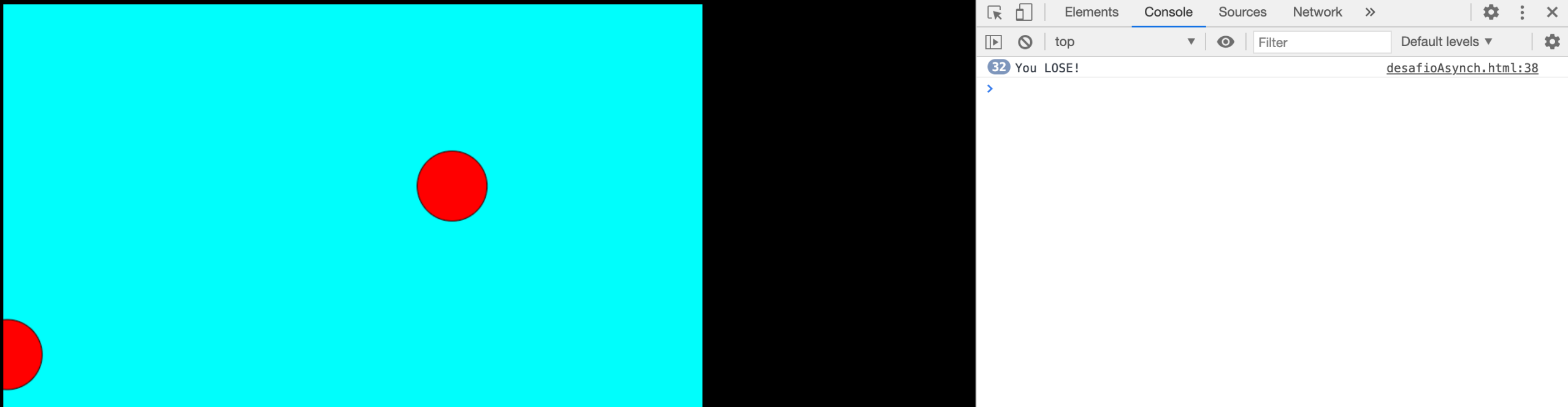
O quadrado da hipotenusa é igual à soma dos quadrados dos catetos. A distância diagonal (variável **dist**) é a hipotenusa. Para a obtermos temos de fazer raiz quadrada (`Math.sqrt`) aos quadrados dos catetos (variáveis **dx** e **dy**).

```
function checkCollision(ball1, ball2) {  
  var dx = ball2.x - ball1.x,  
      dy = ball2.y - ball1.y,  
      dist = Math.sqrt(dx * dx + dy * dy);  
  
  if(dist <= ball1.radius + ball2.radius) {  
    console.log("You LOSE!");  
  }  
}
```



Se a distância for igual ou inferior à soma dos raios, escrevemos para a consola a mensagem "You LOSE!"

Refresh no browser e...



Funciona!

Enunciado

Fazer um mini-jogo em que:

- Temos um **círculo controlado pelos cursores**
- Temos outro **círculo a fazer springing** de um lado para o outro
- O jogador **perde** quando o círculo controlado pelos cursores **colide** com o círculo que está a fazer springing de um lado para o outro.
- O jogador **ganha** se conseguir **sair do canvas** sem acertar no outro círculo.

